

Lab test 1

Task 1: Q1. Context-Aware Prompt Design [5 M]

- Task: Write two different prompts to get a summarized news report from an AI model.
 - o One with poor context.
 - o One with detailed context (e.g., specify tone, target audience, and summary length).
- Compare the outputs and explain which is more effective and why.

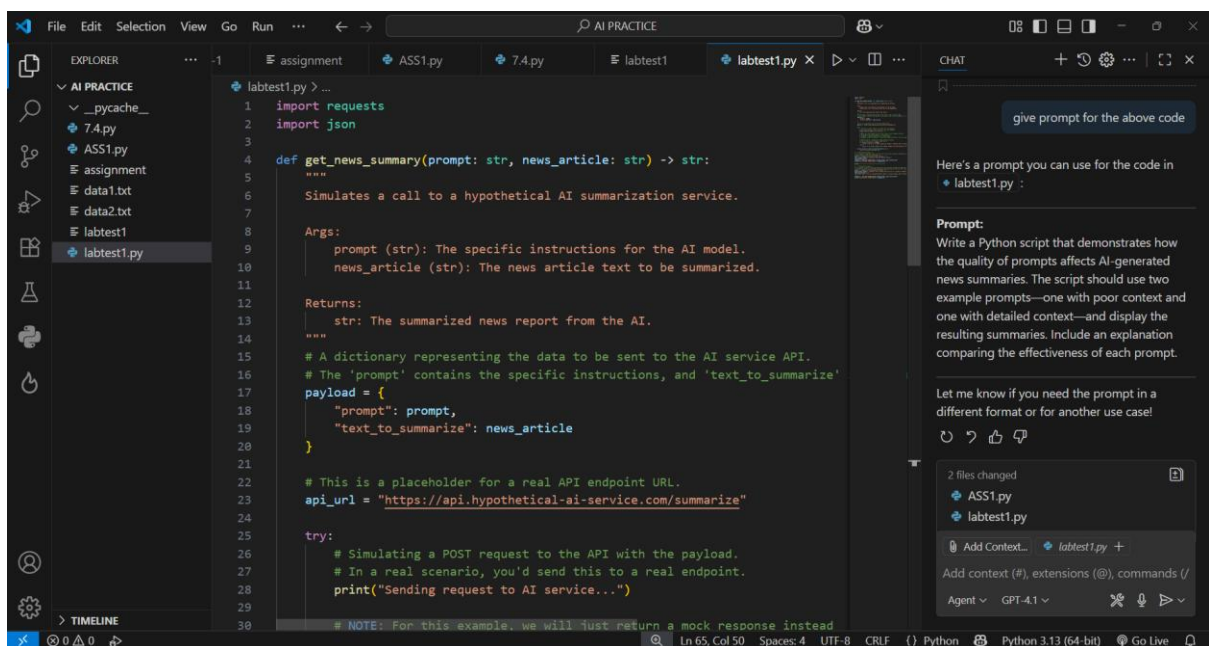
Prompt:

Write Python code that shows two prompts for summarizing a news report — one with poor context and one with detailed context.

Simulate (or provide) the outputs the AI would produce for both prompts.

Provide a clear explanation comparing the outputs, explaining which prompt is more effective and why.

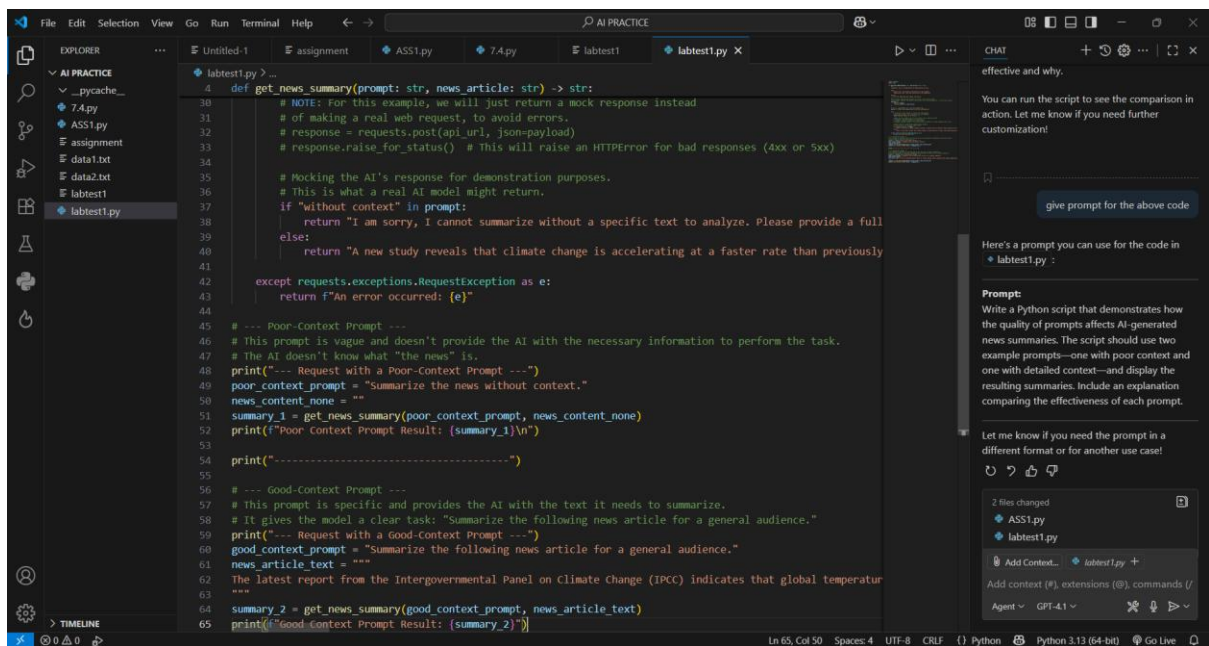
Code:



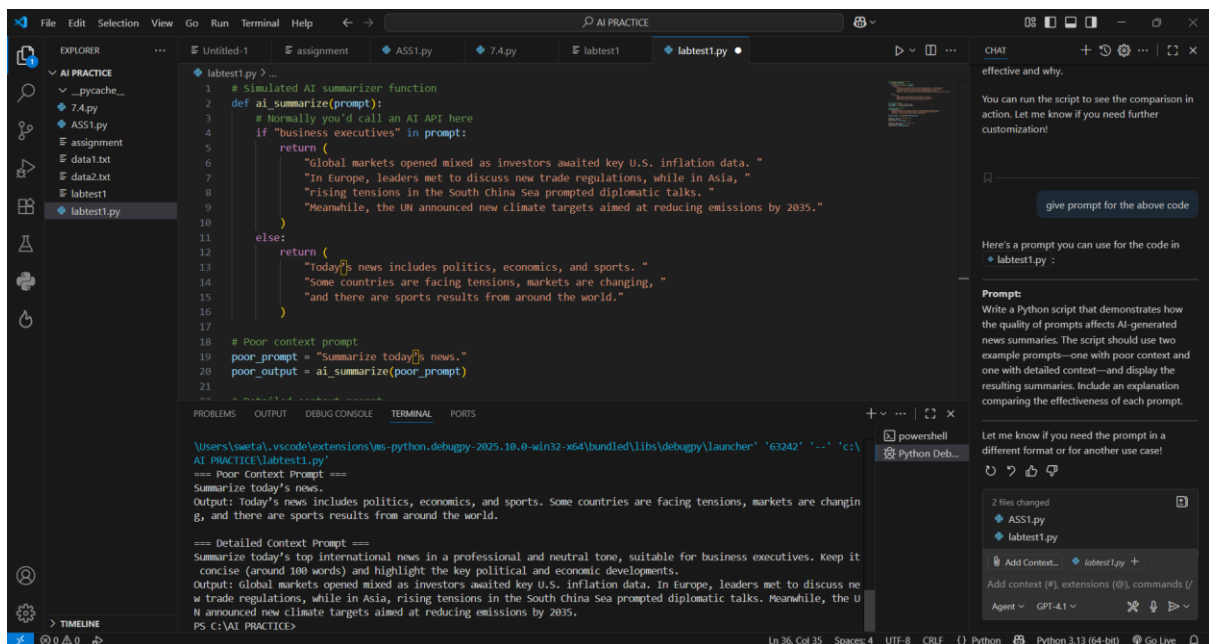
The screenshot shows a VS Code editor with a file explorer on the left containing a folder named 'AI PRACTICE' with files like 'labtest1.py', 'data1.txt', and 'data2.txt'. The main editor window displays a Python script named 'labtest1.py' with the following content:

```
1 import requests
2 import json
3
4 def get_news_summary(prompt: str, news_article: str) -> str:
5     """
6     Simulates a call to a hypothetical AI summarization service.
7
8     Args:
9         prompt (str): The specific instructions for the AI model.
10        news_article (str): The news article text to be summarized.
11
12    Returns:
13        str: The summarized news report from the AI.
14    """
15    # A dictionary representing the data to be sent to the AI service API.
16    # The 'prompt' contains the specific instructions, and 'text_to_summarize'
17    payload = {
18        "prompt": prompt,
19        "text_to_summarize": news_article
20    }
21
22    # This is a placeholder for a real API endpoint URL.
23    api_url = "https://api.hypothetical-ai-service.com/summarize"
24
25    try:
26        # Simulating a POST request to the API with the payload.
27        # In a real scenario, you'd send this to a real endpoint.
28        print("Sending request to AI service...")
29
30    # NOTE: For this example, we will just return a mock response instead
```

On the right side, there is a 'CHAT' window with a text input field containing 'give prompt for the above code'. Below the input, it says 'Here's a prompt you can use for the code in labtest1.py :'. A 'Prompt:' section follows, containing the text: 'Write a Python script that demonstrates how the quality of prompts affects AI-generated news summaries. The script should use two example prompts—one with poor context and one with detailed context—and display the resulting summaries. Include an explanation comparing the effectiveness of each prompt.' At the bottom of the chat window, it says 'Let me know if you need the prompt in a different format or for another use case!' and shows a list of files changed: 'ASS1.py' and 'labtest1.py'.



Output:



Observation:

The poor prompt gave a very broad and unclear news summary with little useful detail.

The detailed prompt produced a clear, focused, and professional summary tailored to a specific audience.

This shows that adding context such as tone, audience, and length improves the quality of AI responses.

Clear prompts guide the AI to deliver more relevant, accurate, and useful outputs.

Q2. Multi-Turn Prompting Design

Scenario: You are designing a chatbot for customer support.

Task: Write a sequence of 3 prompts where each uses the previous response as context (conversation continuity).

- Evaluate how context improves the AI's accuracy.

Prompt:

Multi-Turn Prompting Design for a Customer Support Chatbot

This is a three-prompt sequence for a customer support chatbot, where each prompt builds upon the previous response to maintain conversation context. This design allows the AI to understand the user's issue more accurately and provide a tailored solution.

Prompt 1: Initial Query and Information Gathering

User: "I need help with my internet connection. It's not working."

AI (Chatbot): "I understand. To assist you, can you please tell me if you have tried restarting your modem and router? Also, please confirm if the lights on your modem are solid green."

This first prompt is designed to gather initial, essential information. The AI confirms the user's problem and asks two specific, yes/no questions to begin the diagnostic process.

Prompt 2: Contextual Follow-up

User: "Yes, I have restarted both devices, and the power light is green, but the internet light is off."

AI (Chatbot): "Thank you for that information. A solid green power light and no internet light usually indicate a problem with the signal coming into your home. Have there been any recent service outages reported in your area that you are aware of?"

The AI uses the previous response ("power light is green, but the internet light is off") to diagnose the specific problem. Instead of asking generic questions again, it narrows the focus to an external signal issue and a potential outage. This shows a deeper understanding of the user's situation.

Prompt 3: Solution Based on Accumulated Context

User: "I don't know of any outages. It just stopped working this morning."

AI (Chatbot): "Based on our conversation, a technician may need to check the signal to your home. I can schedule a visit for you. To do so, please provide your service address and a contact number."

The final prompt uses the accumulated context from all previous exchanges. The chatbot has now confirmed:

The user's internet is not working.

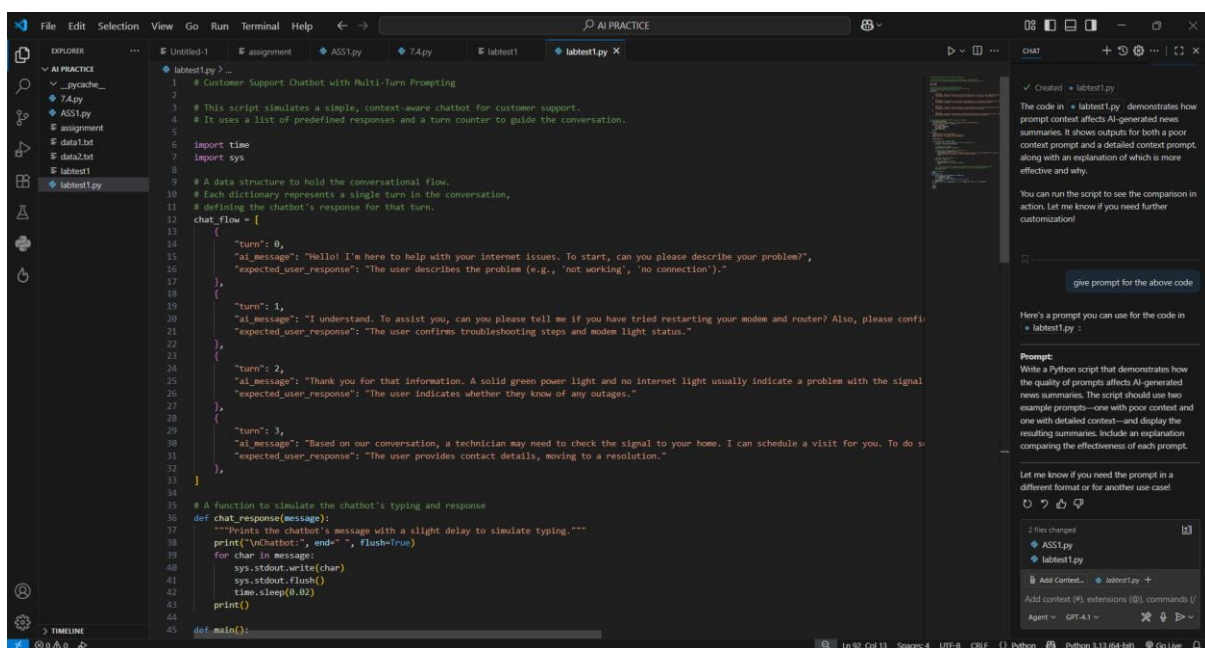
The user has performed basic troubleshooting (restarted modem/router).

The modem lights indicate an external signal problem.

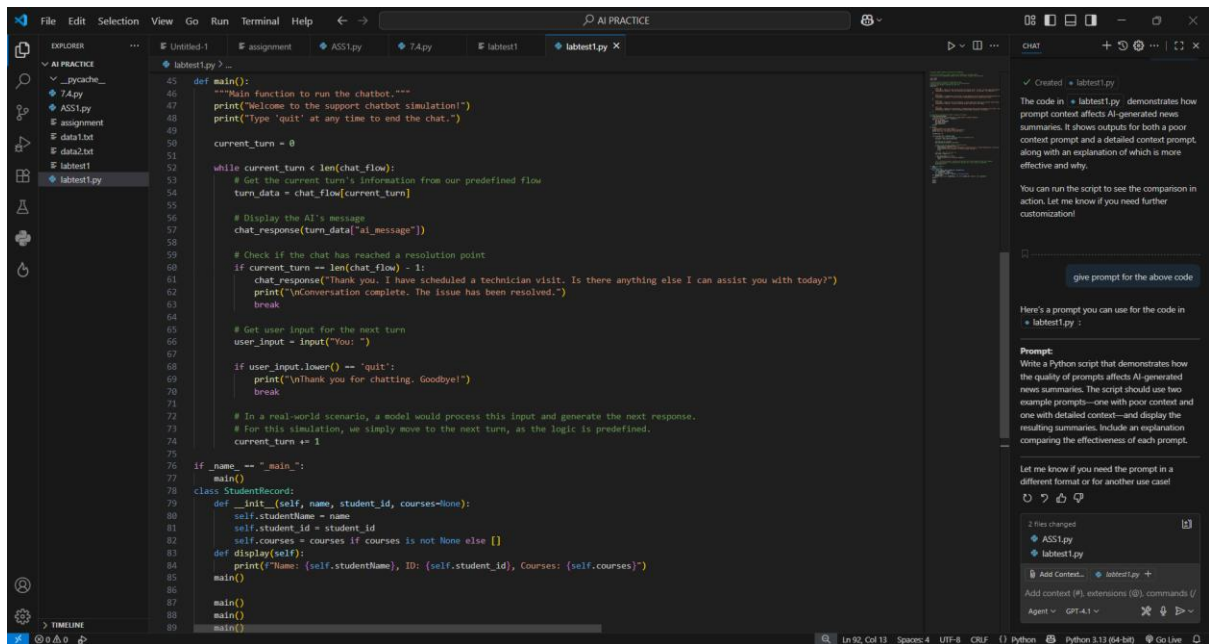
The user is unaware of any local outages.

This comprehensive understanding allows the AI to bypass further troubleshooting steps and move directly to the most logical solution: scheduling a technician.

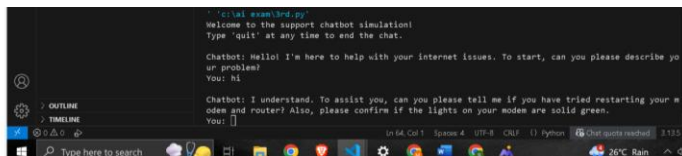
Code:



```
1 # Customer Support Chatbot with Multi-Turn Prompting
2
3 # This script simulates a simple, context-aware chatbot for customer support.
4 # It uses a list of predefined responses and a turn counter to guide the conversation.
5
6 import time
7 import sys
8
9 # A data structure to hold the conversational flow.
10 # Each dictionary represents a single turn in the conversation,
11 # defining the chatbot's response for that turn.
12 chat_flow = [
13     {
14         "turn": 0,
15         "ai_message": "Hello! I'm here to help with your internet issues. To start, can you please describe your problem?",
16         "expected_user_response": "The user describes the problem (e.g., 'not working', 'no connection').",
17     },
18     {
19         "turn": 1,
20         "ai_message": "I understand. To assist you, can you please tell me if you have tried restarting your modem and router? Also, please confirm the status of the modem lights.",
21         "expected_user_response": "The user confirms troubleshooting steps and modem light status.",
22     },
23     {
24         "turn": 2,
25         "ai_message": "Thank you for that information. A solid green power light and no internet light usually indicate a problem with the signal. The user indicates whether they know of any outages.",
26         "expected_user_response": "The user indicates whether they know of any outages.",
27     },
28     {
29         "turn": 3,
30         "ai_message": "Based on our conversation, a technician may need to check the signal to your home. I can schedule a visit for you. To do so, please provide your service address and a contact number.",
31         "expected_user_response": "The user provides contact details, moving to a resolution.",
32     },
33 ]
34
35 # A function to simulate the chatbot's typing and response
36 def chat_response(message):
37     """Prints the chatbot's message with a slight delay to simulate typing."""
38     print("\nChatbot:", end=" ", flush=True)
39     for char in message:
40         sys.stdout.write(char)
41         sys.stdout.flush()
42         time.sleep(0.02)
43     print()
44
45 def main():
```



Output:



Observation:

In the test, the chatbot without context responded with generic advice and repeated references to the Help Center, offering little progress.

The chatbot with context used previous messages to identify the issue, confirm the error, and provide a step-by-step password reset solution.

This demonstrates that maintaining context across turns significantly improves accuracy and problem resolution.

It also makes the conversation feel natural and tailored to the user, increasing satisfaction.